

Problem Description:

The problem that the proposed system is solving is to classify pollen grain images into one of four classes:

1. *Corylus avellana*, well-developed pollen grains (Class 0).
2. *Corylus avellana*, anomalous pollen grains (Class 1).
3. *Alnus*, well-developed pollen grains (Class 2).
4. Debris (bubbles, dust and any non-pollen detected object) (Class 3).

The dataset used is downloaded from the challenge website, and the dataset consists of more than 13 thousand per-object images collected from aerobiological samples, classified into four different categories. Since the test set doesn't have labels, it cannot be used to evaluate the model accuracy, therefore, a 20% of the training set will be used as a testing set, and 10% will be used as a validation set. The precision, recall, and F1 score will be the main evaluation metrics used.

The images shown below in figure 1 are the four images corresponding the available four classes from left to right respectively. The last image to the right which correspond to class 3 is noticeably different from the other three classes. As mentioned in the challenge website, class 3 is not a pollen detected objects.



Figure 1: Example of images from each class (0,1,2,3) from left to right respectively.

The number of images is 13230 images distributed over the four classes. However, the distribution is not uniform as class 2 has over 70% of the training image, class 0 has 15%, and class 1 and class 3 have 15% combined. Images distribution through classes is shown in Table 1.

Table 1: Input images distribution through the four classes

Class Number	Training Images	Testing Images	Total Images
Class 0	1566	-	-
Class 1	733	-	-
Class 2	8216	-	-
Class 3	724	-	-

Total	11239	1991	13230
--------------	--------------	-------------	--------------

Technical Ideas:

Data Balancing to avoid overfitting

Data balance is one of the crucial steps in any machine learning project. Since imbalance data will lead to overfitting in most cases. The data given by the challenge organizer is not well balanced as mentioned earlier and shown in Table 1. For instance, class 2 has over 70% of training images, therefore, if the model classifies any object as class 2, then the model accuracy is 70%. This is a great shortcut for the model while training, and instead of trying hard to generalize the learned model to classify four objects, it is expected that the model will overfit quickly, and output most classification as class 2 if the input data is not balanced. Moreover, the model can ignore class 1 and class 3, and still get training accuracy of 85%. Hence, the input images have been balanced through all classes, each class will have 500 training images, 70 images for validation, and 150 images for testing. The training examples are 70% of the total dataset, validation is 10%, and testing is 20%. The total number of images per class is 720 images, and this is almost the maximum that can be achieved since class 4 has only 724 images.

Table 2: Input images distribution through the four classes after data balancing.


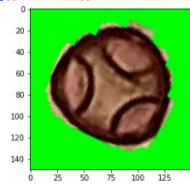

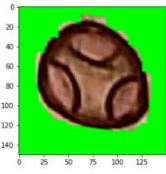

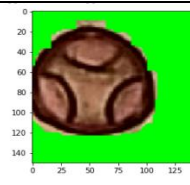

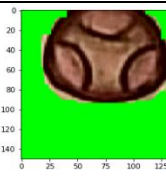
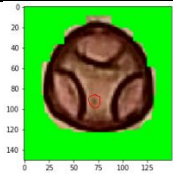
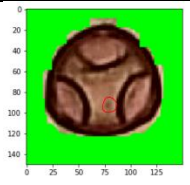

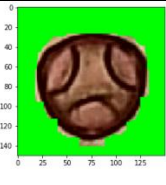
Class Number	Training Images	Validation Images	Testing Images	Total Images
Class 0	500	70	150	720
Class 1	500	70	150	720
Class 2	500	70	150	720
Class 3	500	70	150	720
Total	2000	280	600	2880

Image Augmentation

As the number of images per class is quite low, image augmentation is used to generate more images from the original smaller datasets. There are many image augmentation techniques available, but, in this project, 6 main image transformation techniques are used as shown in Table 3 because of the limited computing resources. The first transformation is rotation, where the input image will be rotated some random degree around its center. Shearing is the second image transformation. Then, horizontal, and vertical shift where the image will be shifted horizontally or vertically some proportion of the image width and height. The last two transformation is the horizontal and vertical flip where the image will be flipped around the x-axis and y-axis, respectively. Image augmentation therefore, can generate infinite number of images since there is infinite combination of the aforementioned transformations, image augmentation is a good solution to avoid overfitting, and increase model accuracy, but it might not lead to the optimum

solution in terms of model accuracy and model loss because in reality the generated images are somewhat similar to the original images.

Table 3: Input images transformations used in the project.

Operation	Original	Transformation	Operation	Original	Transformation
Rotation			Shear		
Width Shift			Height Shift		
Horizontal Flip			Vertical Flip		

Transfer Learning with Resnet50

To solve the problem of not getting the optimum results with image augmentation alone, transfer learning is used. Transfer learning make use of existing pretrained neural networks models, and build on top of these networks simpler networks that will be trained on the smaller datasets while freezing the pretrained layers from being updated in the backpropagation stage. Therefore, the full model will utilize the learned features from the pretrained model, and add to it the learned features that are specific to the smaller datasets. It could be also that the pretrained layers get updated too, and not frozen while training, this is done when the architecture of pretrained networks is more important, and the smaller datasets is far different than the images that these pretrained networks have been trained on.

Resenet50 is one of the most used networks for transfer learning. The motivation of this architecture is the problem of vanishing gradients. As people stared building deeper neural networks to achieve higher accuracies, they noticed that model accuracy started to saturate or even started to get lower as the number of layers increases. They discovered that for extremely deep neural networks, the early layers closer to the input layer is not receiving useful gradients through the backpropagation stage. In order to solve this issue, they added an identity connection or residual connection from each layer to the next layers, but skipping one or more layers down the road. Resnet50 have 5 consecutive convolution blocks, each block has several convolution filters,

then, one average pooling layer, and finally, one fully connected layer. The model is shown in figure 2 below.

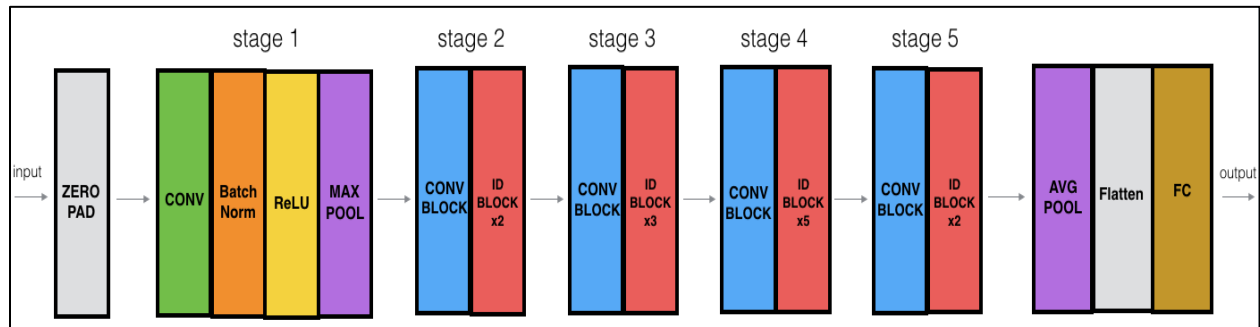


Figure 2: Resnet50 Architecture [1]

Multi Class Classification Problems with Transfer Learning

Multi class classification problems are similar to binary classification problems, but there are some differences. The main difference is in the output layer which will have number of neurons equal to the number of classes instead of having only two neurons. For example, the resnet50 network has 1000 neurons in the last fully connected layer as the network was designed to solve the ImageNet challenge where there were 1000 different objects to classify. Since we have only 4 classes in this project, the last fully connected layer in the resnet50 network is replaced with another fully connected layer with only 4 neurons. Then followed by a SoftMax function to generate the output probabilities for each class. The class with the higher probability is the classification result.

Since the output layer has 4 neurons, the label for the input image is a vector of size 1×4 where the actual image class is set to one, and other classes are set to zero. This is called one hot encoding, and it is used a lot in such problems. The main advantage of one hot encoding is that the network is not going to learn any pattern from the labels. For instance, if the label for each image is set to be from 0 to 3. Then, the network might think that the order of these integers means something so that class 2 is more relevant to class 1 compared to class 3 while the truth is not. Setting all classes to zero resolve this issue. The main drawback for one hot encoding is that the vector become sparse and consume memory storage. For instance, 1000 objects to classify will need a vector is size 1×1000 for each input example which is waste of memory space. But since this project has only four classes, then, one hot encoding is an excellent solution to go with.

Experimental Setup and Results Analysis:

Working Environment

The environment used to train models is Google Collab. It offers Jupiter notebooks and GPU computing resources. Also, it removes the burden of installation of TensorFlow, pytorch and other libraries as well as managing the version of each of these libraries to avoid conflicts. Finally, and I believe the most important, it gives an interface to connect with Google Drive, hence, the dataset can be stored in the cloud, and therefore, the full code and the dataset is online portable and stored in safer place compared to local machines. The deep neural network framework used is Pytorch.

Loss Function

The loss function used while training is the cross-entropy loss, the equation is shown in figure 3 below. As the equation states, for the loss to be zero, the probability for the actual class has to be one, and all other classes probabilities should be zero. This will not happen in practice, but the closer the probabilities to one and zero for the actual class and other classes respectively is better.

$$\text{loss}(x, \text{class}) = -\log \left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])} \right) = -x[\text{class}] + \log \left(\sum_j \exp(x[j]) \right)$$

Figure 3: Loss Function mathematical definition [2].

Optimizer and Learning Rate Scheduler

The optimizer used in the training is the stochastic gradient descent (SGD) with learning rate of 0.001, and momentum of 0.9. the momentum is used to avoid the local minimum problem since the raw stochastic gradient descent might converge to a minimum point, but this point is not the global minimum, but a local minimum.

In addition, a learning rate scheduler is used to decay the learning rate by a factor of 0.1 every 7 epochs. Because when the model is about to converge to the global minimum point, if the learning rate is relatively large, the model will oscillate around the minimum point, but never reach it. To avoid this, the learning rate is lowered to reduce the step size the model performs in each epoch to reach to the global minimum point.

Training and Validation Accuracy and Loss

The model has been trained for 100 epochs, the training and validation loss and accuracy is shown below in figure 4. Because of transfer learning, the model loss has dropped very fast in the first 20 epochs, and almost saturated at that point. For the next 75 epochs, no considerable change in then loss and accuracy. The good thing is that the training and validation are synchronized with each other which indicate that the model has learned the required features and is able to generalize. The maximum model validation accuracy is 86%, and the minimum validation loss is 0.4.

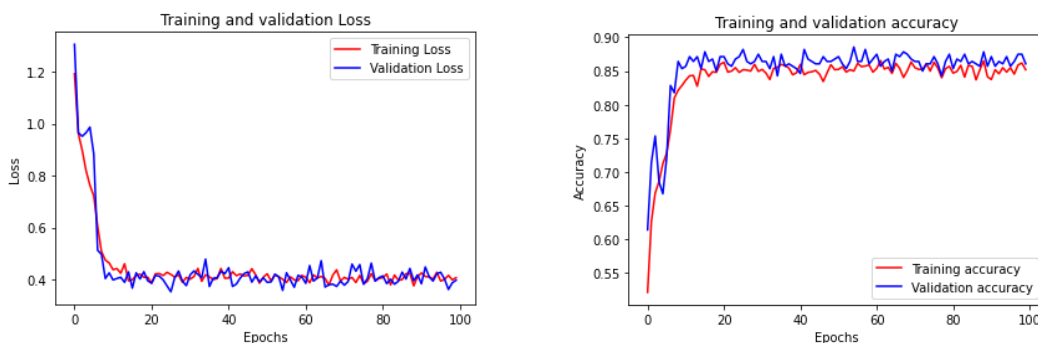


Figure 4: Training and Validation Loss and Accuracy.

Recall, Precision, and F1 Score

Precision and recall are better representative of the model performance compared to accuracy. Precision is the ratio of correctly predicted positive observation to the total predicted positive observation. For instance, for class 1, the precision is the lowest compared to other classes, this is because the network kind of focused a little bit more on class 1 which result in class 1 being the best classified class. This resulted in false prediction of images corresponding to other classes to into class 1. Recall is ratio of correctly predicted positive observation to all observation in the actual class, and class 1 has the highest value of 0.93. This explains why the precision of class 1 is the lowest. The network is favoring class 1 in its output.

Class 3 is the opposite of class 1 since it has the highest precision and lowest recall. This indicate that the network is favoring class 3 less compared to others, which resulted in less false positive instances for class 3 which is good as the network is not outputting class 3 as solution in most cases, but the number of false negative which are the cases where the network misclassifies an actual class 3 image as other classes.

Class 0 and class 2 are somewhat similar as their precision and recall values are close to each other. However, class 0 seems to be performing better as the precision value is higher which is reflected in higher F1 score compared to class 2.

The precision and recall described earlier are calculated based on the confusion matrix. The value of true positive for each class are reported on the diagonal where class 1 has the highest value of 139 out of 150 test images. Class 3 is the lowest with true positive of 122 out of 150 test images. If you sum the values across the vertical axis, you will get the false positive values for each class. For instance, class 1 has 29 false positive instances, therefore, it has the minimum precision. If the values across the horizontal axis are summed, we will get the false negative where class 3 has the highest value of 28 instances. The precision, recall, f1 score and confusion matrix are all shown in figure 5 below. The testing accuracy as shown is 86%.

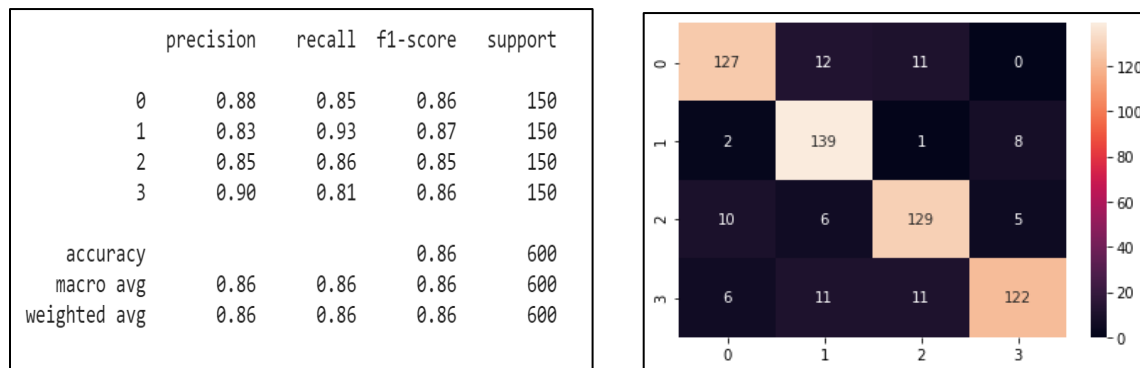


Figure 5: Precision and Recall values for Testing Images (Left), and confusion matrix to the right.

Future Direction and Summary

The results obtained are quite good where the testing accuracy is 86%. But to push the accuracy more, we need to use other techniques. As shown in figure 4, the validation accuracy saturated indicating that with transfer learning on this dataset, the accuracy cannot increase anymore. Another solution that might increase the model accuracy is to use Siamese network to find the distance metric between each class and other classes.

A Siamese neural network is a special type of neural network that shares the neural network weights across two input vectors. During training, the network gets two inputs for each instance, forward propagate the two inputs through the network, calculate the difference between the two outputs, and update the weights using backpropagation. The network has only one weight matrix that is being shared across the two inputs. This architecture helps the neural network learn a distance function or a similarity between the two inputs. This helps a lot in cases where the number of training examples of some classes in the dataset is small. For instance, if we have two classes where each class has 500 images, then we can generate 250000 similar pair of images for each class, and 250000 pair of dissimilar pair of images. Then, we can ask the network to find the distance function that will minimize the distance between similar pairs, and maximize the distance between dissimilar pairs.

For this project, this might lead to an ensembled model consisting of four of these networks, each trying to minimize the distance for one class while trying to maximize the distance with other classes. Then, one last layer that will consider the output of these four Siamese networks, and output probabilities score for each class.

References.

- [1] <https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33>
- [2] <https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>